BAB II
LANDASAN TEORI

2.1. Tinjauan Pustaka

Sebelum melakukan penelitian, peneliti akan melakukan studi literature terhadap penelitian yang sudah dilakukan serta memiliki beberapa keterkaitan dengan penelitian yang lainnya.

Pada penelitian yang dilakukan oleh Rikie Kartadie, yaitu Mikrotik RB750 Routerboard Sebagai Alternatif Switch *OpenFlow* Software-base. Pada penelitian ini pengujian dilakukan tehadap Mikrotik Routerboard RB750 yang telah ditambahkan *OpenFlow agent* agar bisa diimplementasikan pada arsitektur SDN/*OpenFlow*. Setelah itu dari rancangan prototipe tersebut dibandingkan dengan simulasi menggunakan emulator *mininet* sebagai data pembanding (dianggap sebagai representative dari *hardware-base switch*) dengan topologi jaringan yang sama. Hasil yang diperoleh dari penelitian tersebut adalah dari uji *latency* prototipe memberikan hasil yang lebih tinggi dari pada pebandingnya, untuk throughput dan jitter prototipe memberikan nilai yang lebih rendah dari pembandingnya. Walaupun prototipe memberikan nilai perfoma yang masih rendah, prototipe ini bisa jadi alternatif pengganti untuk switch *OpenFlow software-base.* [6]

Penelitian lainnya yang dilakukan oleh Shiva Rowshanrad tentang *Perfomance Evaluation of SDN Controller: Floodlight and OpenDaylight.* Di penelitian ini dibandingkan antara kontroler Floodlight dengan OpenDaylight yang mana akan berfungsi dengan baik pada kondisi tertentu. Untuk parameter pengujiannya menggunakan nilai *latency, jitter, packet loss* dan *throughput* dengan tiga jenis topologi yang berbeda yaitu topologi *single*, topologi *linear* dan topologo *tree*. Dengan hasil 95% keakuratan menyatakan bahwa OpenDaylight lebih unggul dari pada Floodlight dalam kondisi beban rendah dan pada topologi *tree* dengan beban sedang dengan nilai parameter *latency*. Berbeda dengan kondisi beban berat pada topologi *tree* dengan nilai parameter *packet loss*, Floodlight lebih unggul dari OpenDaylight dalam hal tersebut.

Dari hasil tersebut tidak perbedaam yang signifikan dari kedua kontroler tersebut dalam berbagai hal.[7]

Dari penelitian yang telah di jelaskan tersebut hanya melakukan pengujian pada infrastruktur SDN dalam ruang linkup simulasi yang di anggap sebagai representative dari jaringan yang sebernarnya. Untuk selanjutnya penelitian ini akan melakukan implementasi infrastruktur SDN pada jaringan laboratorium teknik informatika yang sebelumnya belum menggunakan infrastruktur tersebut. Setelah melakukan implementasi selanjutnya akan dilakukan pengujian dan analisa perfomansi dari segi *Quality of Service* (QoS) dan akan dibandingkan sebelum dengan sesudah di implementasikan infrastruktur SDN menggunakan parameter *latency*, *throughput*, *jitter*, *bandwidth* dan *packet loss.*

2.2. Software Defined Network

Software Defined Network adalah sebuah paradigma baru dalam aritektur jaringan komputer, yang memiliki karakteristik dinamis, *manageable*, *cost-effective*, dan *adaptable*, sehingga sangat ideal untuk kebutuhan jaringan yang bersifat dinamis dan high-bandwidth. Arsitektur SDN ini memisahkan antara network control dan fungsi forwarding, sehingga network control tersebut menjadi *directly programmable* (dapat diprogram secara langsung), sedangkan infrastruktur yang mendasarinya dapat diabstraksikan untuk layer aplikasi dan network services.

Menurut Eueung Mulyana dkk *Software Defined Network*, adalah istilah yang merujuk pada konsep/paradigma baru dalam mendesain, mengelola dan mengimplementasikan jaringan, terutama untuk mendukung kebutuhan dan inovasi di bidang ini yg semakin lama semakin kompleks. Konsep dasar SDN adalah dengan melakukan pemisahan eksplisit antara control dan forwarding plane, serta kemudian melakukan abstraksi sistem dan meng-isolasi kompleksitas yg ada pada komponen atau sub-sistem dengan mendefinisikan antar-muka (interface) yg standard.[8]

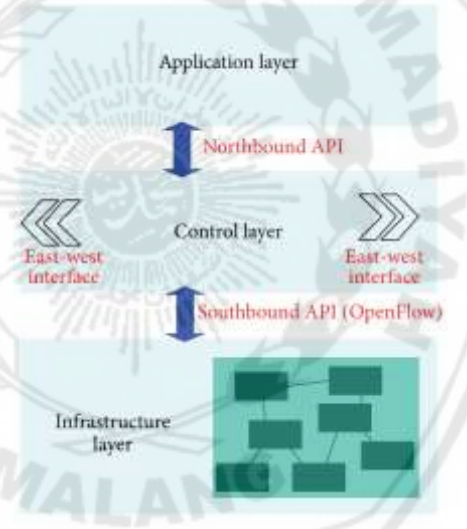
Beberapa aspek penting dari SDN adalah[8] :

1. Adanya pemisahan secara fisik/eksplisit antara *forwarding / data-plane* dan *control-plane*.

2. Antarmuka standard (vendor-agnosic) untuk memprogram perangkat jaringan.
3. *Control-plane* yang terpusat (secara logika) atau adanya sistem operasi jaringan yang mampu membentuk peta logika *(logical map)* dari seluruh jaringan dan kemudian memrepresentasikannya melalui (sejenis) *API (Application Programming Interface)*
4. Virtualisasi dimana beberapa sistem operasi jaringan dapat mengkontrol bagian-bagian *(slices atau substrates)* dari perangkat yang sama.

2.2.1. Arsitektur Software Defined Network

Dalam konsep SDN, terdapat 3 layer utama yang menyusun arsitekture dari jaringan SDN, yaitu *Application Layer, Control Layer,* dan *Infrastructure Layer*. Seperti Gambar 2.1.



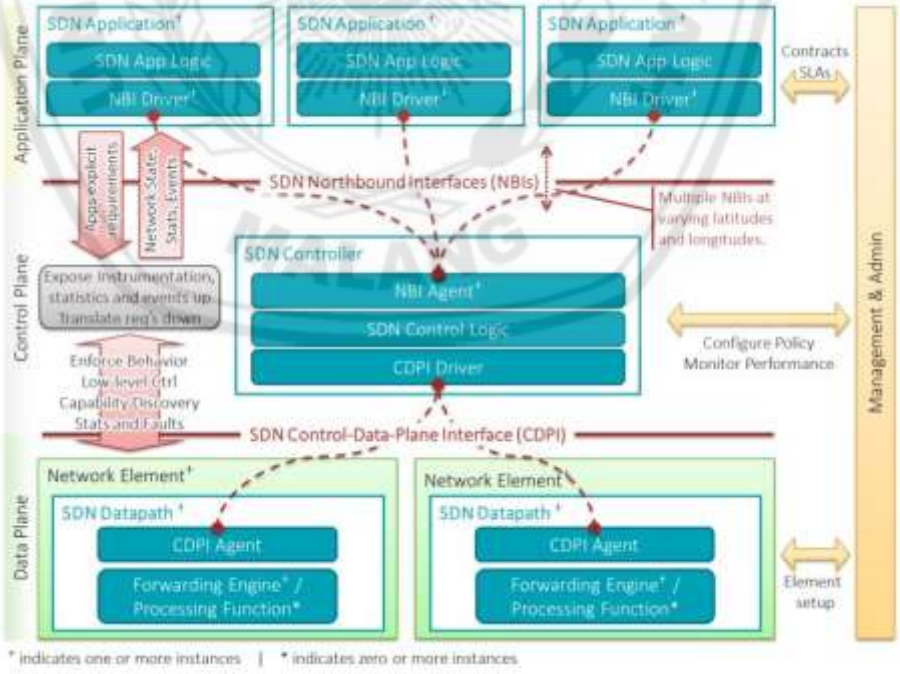
Gambar 2. 1 Arsitektur Software Defined Network [9]

- *Application Layer* adalah layer yang terdiri dari proses bisnis dari *end-user* untuk layanan komunikasi dengan SDN. Antara *application layer* dengan *control layer* dibatasi oleh sebuah *Northbound* API.[9]
- *Control Layer* pada layer ini menyediakan fungsi control terhadap *forwarding* pada jaringan melalui sebuah *open interface*.[9]

- *Infrastructure Layer* yang mana pada bagian ini adalah terdiri perangkat jaringan secara fisik yang dapan menyediakan *packet switching* dan *forwarding*.[9]

2.2.2. Komunikasi Komponen Software Defined Network

Pada arsitektur ini jaringan dilihat sebagai data, kontrol dan aplikasi, dari **Gambar 2.2** dibagi menjadi 3 area utama yaitu bagian atas, bawah dan tengah. Di bagian bawah terdapat *data plane* yang terdiri dari elemen jaringan, yang mana SDN *datapath* dapat bekomunikasi melalui *SDN Control-Data-Plane Interface (CDPI) Agent*. Di bagian atas, Aplikasi SDN / antar muka berada pada bagian *Application Plane*, dan untuk dapat saling berkomunikasi melalui *NorthBound Interface (NBI) Drivers*. Dan pada bagian tengah, *SDN Controller* menterjemahkan keperluan bagian bawah dan dan memberikan kontol tingkat rendah atas SDN *datapath*, *controller* juga bertugas memberikan informasi mengenai bagian bawah ke Aplikasi SDN / antarmuka. *Management and Admin* betanggung jawab untuk meyiapkan elemen jaringan, menetapkan datapath terhadap controller. Bagian ini juga bertanggung jawab untuk melakukan konfigurasi terhadap perangkat SDN.[10]



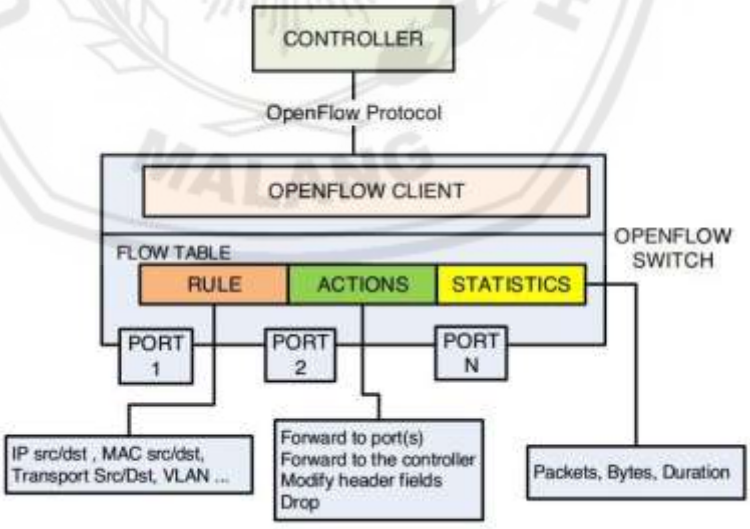
Gambar 2. 2 Komponen Arsitektur dan Interaksi Komponen SDN[10]

2.3. OpenFlow

OpenFlow adalah suatu protokol bersifat terbuka yang menghubungkan antara perangkat jaringan dengan sebuah kontroler, yang memungkinkan untuk melakukan memprograman *data plane* pada *switch* atau *router* dan dapat mengontrol secara langsung lalu lintas paket pada *forwarding plane.* [2] Dengan OpenFlow di tempatkan pada sebuah switch atau router maka dapat dilakukan flow forwarding berbasis *network layer* dan juga dapat melakukan pengaturan pergerakan paket secara terpusat mulai dari layer 2 sampai layer 7 forwading *(flow granuality)*, sehingga aliran paket di jaringan dapat diprogram secara independen. Hal ini dapat dilakukan dengan membuat algoritma dan *forwarding rules* di *controller* kemudian aturan tersebut didistribusikan ke *switch* dan *router* yang telah menggunakan protokol ini.

2.3.1. OpenFlow Switch

Setelah Openflow dimasukkan ke perangkat switch atau router maka perangkat tersebut telah menjadi OpenFlow Switch atau Forwarding Device. OpenFlow Switch terdiri dari satu atau lebih flow table dan sebuah layer abstrak yang digunakan untuk berkomunikasi dengan sebuah kontroler melalui protokol openflow. [11]



Gambar 2. 3 Arsitektur OpenFlow Switch [11]

2.3.2. Flow Table

Flow Table adalah suatu tabel yang berisi entries, dimana masing-masing *flow entries* ini menentukan bagaimana paket-paket diubah menjadi suatu aliran *(flow)* yang akan diproses dan kemudian di teruskan. Pada Gambar 2.3 adalah komponen yang terdapat di dalam flow tabel.

Flow entries terdiri dari :[11]

- *Rule (maching rules)*, berfungsi untuk mencocokan paket yang masuk dan biasanya informasi header paket, ingress port (jalan masuk), dan metadata.
- *Statistics* berfungsi untuk mengumpulkan data yang digunakan untuk aliran (flow) khusus seperti jumlah paket yang diterima, kapasitas dan durasi dari suatu aliran.
- *Actions* adalah suatu set yang berisi intruksi atau tindakan, yang kemudian akan diterapkan pada sebuah match dan juga mengatur bagaimana cara mencocokan paket-paket tersebut.

2.3.3. Switch OpenFlow Software-base

Switch OpenFlow Software-base adalah perangkat *switch* berdasarkan perngkat lunak yang menggunakan sistem operasi UNIX / Linux untuk pengimplementasian seluruh fungsi switch *OpenFlow software-based*, vendor biasanya menyediakan *package* tambahan pada untuk perangkatnya. Jenis switch ini paling sering digunakan pada ruang lingkup pengujian dan pengembangan aplikasi jaringan berbasis openflow. [4] Untuk pengimplementasian pada Routerboard MikroTik tidak perlu mengganti OS atau memodifikasinya karena MikroTik menyediakan *package OpenFlow* tambahan untuk RouterOS yang siap untuk di implementasikan. Setelah ditambahkan *package* tersebut Routerboard MikroTik akan menjadi *switch OpenFlow Software-base.*

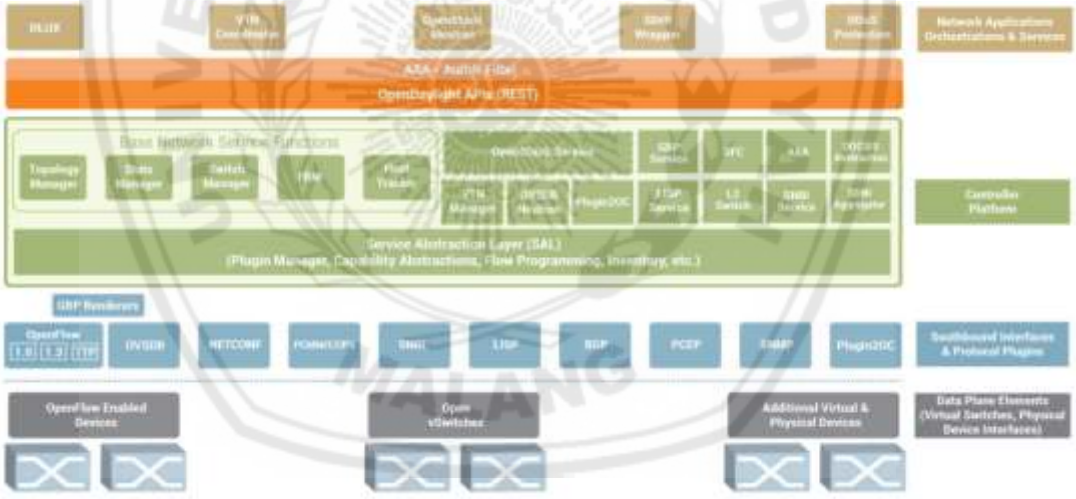
2.4. OpenDaylight

OpenDaylight adalah sebuah proyek yang terbuka kolaboratif di Linux Foundation yang bertujuan untuk penerapan *Software Define Network (SDN)* dan *Network Function Virtualization (NFV)* untuk jaringan di setiap ukuran dan skala yang memiliki kemampuan *programmability*.

Opendaylight Controller adalah sebuah proyek yang terbuka dengan kontroler yang bersifat modular, dapat dipasangkan dan fleksibel. Kontroler ini berjalan di atas *Java Virtual Machine (JVM)* dan di implementasikan pada perangkat lunak. Dengan demikian OpenDaylight dapat di gunakan pada system operasi yang mendukung java. OpenDaylight menggunakan *open northbound API* yang ada pada aplikasi untuk berkomukasi. Algoritma dan proses bisnis dijalankan pada aplikasi, sedangkan kontroler digunakan untuk mengumpulkan data jaringan, menjalankan algoritma analisis dan selanjutnya menggunakan kontroler untuk menerapkan aturan baru. Pada bagian *Southbount Interface* sudah mendukung beberapa protokol yang secara terpisah misalnya OpenFlow 1.0, OpenFlow 1.3, BGP-LS dan lain-lain. [8]

2.4.1. Arsitektur OpenDaylight

Untuk komponen-komponen yang ada pada arsitektur OpenDaylight dapat dilihat pada **Gambar 2.4**



Gambar 2. 4 Arsitektur OpenDaylight[12]

Untuk penjelasan dari fungsi-fungsi dari Gambar 2.4 sebagai berikut[12]:

2.4.2. Controller Platform

Controler Platform ini mewakili bagian utama dari arsitektur SDN. Lapisan ini menggunakan API Northbound yang menghubungkan ke aplikasi

jaringan untuk mengendalikan dan mengelola elemen fisik dan virtual di dalam jaringan. Bagian ini terdiri dari Base Network Service Functions (BNSFs), Platform Network Service Functions dan Service Abstraction Layer (SAL).

a. *Base Network Service Functions (*BNSF) bertanggung jawab untuk memperoleh informasi statistik tentang elemen-elemen jaringan dan fungsi dari elemen-elemen tersebut. Bagian ini menggunakan API Northbound untuk memperoleh informasi dan statistik. Di dalam bagian ini terdapat bagian-bagiannya :
   - *Topology Manager,* menyimpan informasi tentang topologi, switch. Bagian ini memberitahukan ketika ada penambahan atau pengurangan switch pada topologi jaringan. Untuk mengakses topologi manager dapat melalui *API Northbound*.
   - *Statistics Manager,* mengumpulkan informasi dari switch yang di kelola. Informasi tersebut berupa port, tabel dan aliran data yang melalui switch.
   - *Switch Manager*, menyimpan detail mengenai switch dan port yang ada pada switch untuk di identifikasi, dan menyimpannya pada *data tree*.
   - *Forwarding Rules Manager* (FRM), melakukan pengercekan pada *flow*, menangani apabila konflik dan memvalidasinya. Menyediakan aturan dasar *forwarding* seperti *OF rules* dan menerapkannya pada switch yang di kelola melalui *Southbound interface*.
   - *Inventory Manager,* memelihara database untuk daftar switch yg terhubung.
   - *Host Tracker,* melakukan pelacakan host terakhir yang terhubung ke jaringan (switch, port dan koneksi ke) dan menyimpan informasi berupa alamat MAC, alamat jaringan, tipe switch dan tipe port. Dengan database dinamis yang menggunakan alamat MAC dari host.

b. *Platform Network Service Functions:* kontroler OpenDaylight berisi service yang bersifat pluggable yang bertugas pada jaringan dan ekstensi khusus lainnya untuk meningkatkan fungsuonalitas SDN. Di dalam bagian ini terdapat bagian-bagiannya :

- *Affinity Metadata Service*, menyediakan Northbound API secara langsung untuk keperluan jaringan pada aplikasi dan mengkomunikasikan beban pada kontroler.
- *Virtual Tenant Network* (VTN) Manager, membuat dan mengelola sebuah *multi-tenant* pada jaringan virtual. VTN mengijinkan pengguna untuk merancang sebuah jaringan yang logic, terlepas dari jaringan fisiknya.
- *L2 Switch*, menyediakan fungsi dari L2 switch dan membuat beberapa sevice dapat digunakan kembali seperti *address tracking, basic spanning tree protocol,* dan perhitungan jalur optimal.
- *Service Function Chaining* (SFC), menyediakan kemampuan untuk menentukan *chain* pada layanan jaringan seperti *firewall, router* dan *load balancing* untuk memastikan adanya layanan untuk lalu lintas data.
- *Group-Based Policy* (GBP) memisahkan antara keperluan konektivitas aplikasi dengan rincian elemen jaringan melalui *application-focused policy model*. Bagian ini juga mengklasifikasikan *end-point* ke dalam grup berdasarkan keperluan aplikasi dan menerapakan kebijakan/*policy* kepada grup-grup.
- *Authentication, Authorization, and Accounting* (AAA) *Service*, untuk menyediakan fungsi umum dari AAA pada OpenDaylight. Identitas pengguna dan perangkat diverifikasi menggunakan token/ *claim-based authentication.* Ijin untuk menggakses *resource,* seperti *RPC,* notifikasi dan *subset data tree* diperiksa bedasarkan *role-based access control* (RBAC). Bagian ini juga mencatat semua akses *resource* untuk tujuan analisi dan audit keamanan.

c. Service Abstraction Layer (SAL) : adalah bagian utama dari OpenDaylight, SAL memungkinkan OpenDaylight untuk mendukung beberapa protokol Southbound melalui plugin itu sendiri dan menyediakan seperangkat layanan/*service* yang sama kepada aplikasi. *Device Discovery* adalah layanan yang disediakan oleh SAL yang diberikan ke *Topologi Manager* untuk membuat topologi jaringan. Sebagian besar dari layanan SAL dibuat

berdasarkan fitur dari plugin Southbound. Permintaan suatu layanan akan diberikan oleh SAL secara penuh kepada sebuah switch, terlepasa dari protokol Southbound yang digunkan oleh switch. Plugin dari Northbound dan Southbound bisa menjadi penyedia atau pemakai layanan atau bahkan keduanya. SAL dapat berfungsi sebagai *registry* dari layanan/ *service* dari API. Ketika konsumen dari layanan tersebut meminta API umum, maka SAL yang akan menghubungkan antara penyedia layanan dan konsumen.

2.4.3. Southbound Interface dan Protocols Plugins

Untuk memungkinkan komunikasi yang aman antara kontroler dengan perngkat jaringan, maka protokol Southbound yang digunakan. Perangkat jaringan dapat di atur, dikonfigurasi dan dimonitoring melalui protokol ini. Melalui protokol ini OpenDaylight dapat mendukung jaringan yang beragam.

- OpenFlow Plugin menerapkan spesifikasi dari protokol OpenFlow yang telah dikembangkan, dan memungkinkan untuk mendukung OpenFlow versi 1.0, 1.3 dan *Table Type Patterns (TTPs)* yang mana memungkinkan kontroler OpenFlow dan switch Openflow dapat berkomunikasi.
- Open vSwtich Database (OVSDB) Plugin adalah model dari OVSDB protokol yang mengatur dari *open virtual switch.* OVSDB ini kembangkan pada *firmware* switch-switch yang terbaru.
- SNMP Plugin melakukam pengembangan dari *Simple Network Management Protocol (SNMP)* pada protokol Southbound untuk mengatur port Ethernet pada switch. Konfigurasi aliran data diatur melalui *forwarding table, ACL, and VLAN table*.
- BGP-LS/PCEP Plugins diimplementasikan dengan *Java-based Border Gateway Protocol (BGP)* dan *Path Computation Element Protocol (PCEP)* untuk mendukung *BGP Linkstate Distribution* di OpenDaylight. Plugin ini juga sebagai informasi bagi L3 sedangkan PCEP digunkan untuk membuat jalur ke semua jaringan.

- Network Configuration Protocol (NETCONF) Plugin dikembangkan pada OpenDaylight untuk mengelola dan konfigurasi pada perangkat jaringan yang mendukung protokol NETCONF.

2.4.4. Network Application dan Services

Untuk memungkinkan komunikasi yang aman antara kontroler dengan perngkat jaringan, maka protokol Southbound yang digunakan. Perangkat jaringan dapat di atur, dikonfigurasi dan dimonitoring melalui protokol ini. Melalui protokol ini OpenDaylight dapat mendukung jaringan yang beragam.

- *OpenFlow Plugin* menerapkan spesifikasi dari protokol OpenFlow yang telah dikembangkan, dan memungkinkan untuk mendukung OpenFlow versi 1.0, 1.3 dan *Table Type Patterns (TTPs)* yang mana memungkinkan kontroler OpenFlow dan switch Openflow dapat berkomunikasi.
- *Open vSwtich Database (OVSDB) Plugin* adalah model dari OVSDB protokol yang mengatur dari *open virtual switch.* OVSDB ini kembangkan pada *firmware* switch-switch yang terbaru.
- *SNMP Plugin* melakukam pengembangan dari *Simple Network Management Protocol (SNMP)* pada protokol Southbound untuk mengatur port Ethernet pada switch. Konfigurasi aliran data diatur melalui *forwarding table, ACL, and VLAN table*.
- *BGP-LS/PCEP Plugins* diimplementasikan dengan *Java-based Border Gateway Protocol (BGP)* dan *Path Computation Element Protocol (PCEP)* untuk mendukung *BGP Linkstate Distribution* di OpenDaylight. Plugin ini juga sebagai informasi bagi L3 sedangkan PCEP digunkan untuk membuat jalur ke semua jaringan.
- *Network Configuration Protocol* (NETCONF) *Plugin* dikembangkan pada OpenDaylight untuk mengelola dan konfigurasi pada perangkat jaringan yang mendukung protokol NETCONF.

2.5. Quality of Service

Quality of Service (QoS) atau kualitas suatu layanan adalah kemampuan jaringan untuk menyediakan suatu layanan yang baik dengan menggunkan parameter-parameter tertentu. Parameter dari QoS adalah *latency, jitter, packet loss, throughput* dan *bandwidth*. Berikut adalah penjelasan tentang parameter-parameter tersebut[13]:

2.5.1. Latency / Delay

Latency atau Delay merupakan satuan waktu yang dibutuhkan data untuk sampai dari tempat asal ke tujuan. *Latency* dapat dipengaruhi oleh media fisik, perngkat yang digunakan, jarak dan waktu proses dari data tersebut.

Tabel 2. 1 Kategori Latency[13]

Kategori Latensi	Besar Delay (ms)
Sangat Bagus	< 150 ms
Bagus	150 ms s/d 300 ms
Sedang	300 ms s/d 450 ms
Jelek	> 450 ms

2.5.2. Jitter

Jitter atau variasi delay merupakan variasi pengiriman antar paket yang berurutan terjadi pada jaringan internet. Jitter dapat dipengaruhi oleh beban yang ada pada jaringan dan besarnya tumbukan antar paket yang ada. Jitter dapat dihitung dari mengambil nilai rata-rata deviasi dari selisih Delay antar paket yang dikirim.

Tabel 2. 2 Kategori Jitter [13]

Kategori Jitter	Jitter (ms)
Sangat Bagus	0 ms
Bagus	0 ms s/d 75 ms
Sedang	75 ms s/d 125 ms
Jelek	125 ms s/d 225 ms

2.5.3. Packet Loss

Packet Loss adalah merupakan parameter yang menunjukkan jumlah total paket yang hilang selama pengiriman melalui jaringan IP. Hilangnya biasanya

disebabkan oleh *network congestion*. Packet Loss dihitung dengan persentasi paket yang hilang dengan jumlah paket yang dikirim.

Tabel 2. 3 Kategori Packet Loss [13]

Kategori Degredasi	Packet Loss (%)
Sangat Bagus	0
Bagus	3
Sedang	15
Jelek	24

2.5.4. Troughtput

Throughput merupakan kecepatan transfer data yang sebenarnya, diukur dalam *bit per second* (bps). Throughput dihitung dengan jumlah total kedatangan paket yang diterima pada tujuan selama waktu tertentu dibagi dengan durasi interval waktu tertentu.

Tabel 2. 4 Kategori Throughput [13]

Kategori Throughput	Throughput (bps)
Sangat Bagus	100
Bagus	75
Sedang	50
Jelek	< 25

2.5.5. Bandwidth

Bandwidth adalah banyaknya data yang dapat dikirim atau diterima antar perangkat komputer, satuan bandwidth adalah dalam *bit per second* (bps), kecepatan dari bandwidth dipengaruhi oleh media yang digunakan, perangkat jaringan yang digunakan. Untuk bandwidth semakin cepat semakin bagus, karena semakin banyak data yang dikirim atau diterima maka semakin bagus.